

**Version 5.5**

**02/11/2016**

# Contents

---

Current DMDC IT Environment .....	3
.....	4
.....	4
Section 1 – DMDC Hardware Environment.....	7
Linux\Solaris.....	8
Section 2 – DMDC Software Environment.....	8
Linux\Solaris.....	8
Section 3 – DMDC Application Standards .....	8
Linux\Solaris.....	8
Mainframe .....	10
Section 4 - DMDC Data Access Standards.....	10
Linux\Solaris.....	10
Section 5 - DMDC Data Store Standards .....	15
Database tools .....	15
Linux\Solaris.....	15
Data Store Definitions:.....	16
Data Architecture Requirements Model:.....	18
Data Architecture Use Cases:.....	19
Data Collection.....	19
Data Storage.....	19
Data Usage .....	19
Data Architecture Requirements: .....	21
Data Collection.....	21
Data Storage.....	21
Data Usage .....	22

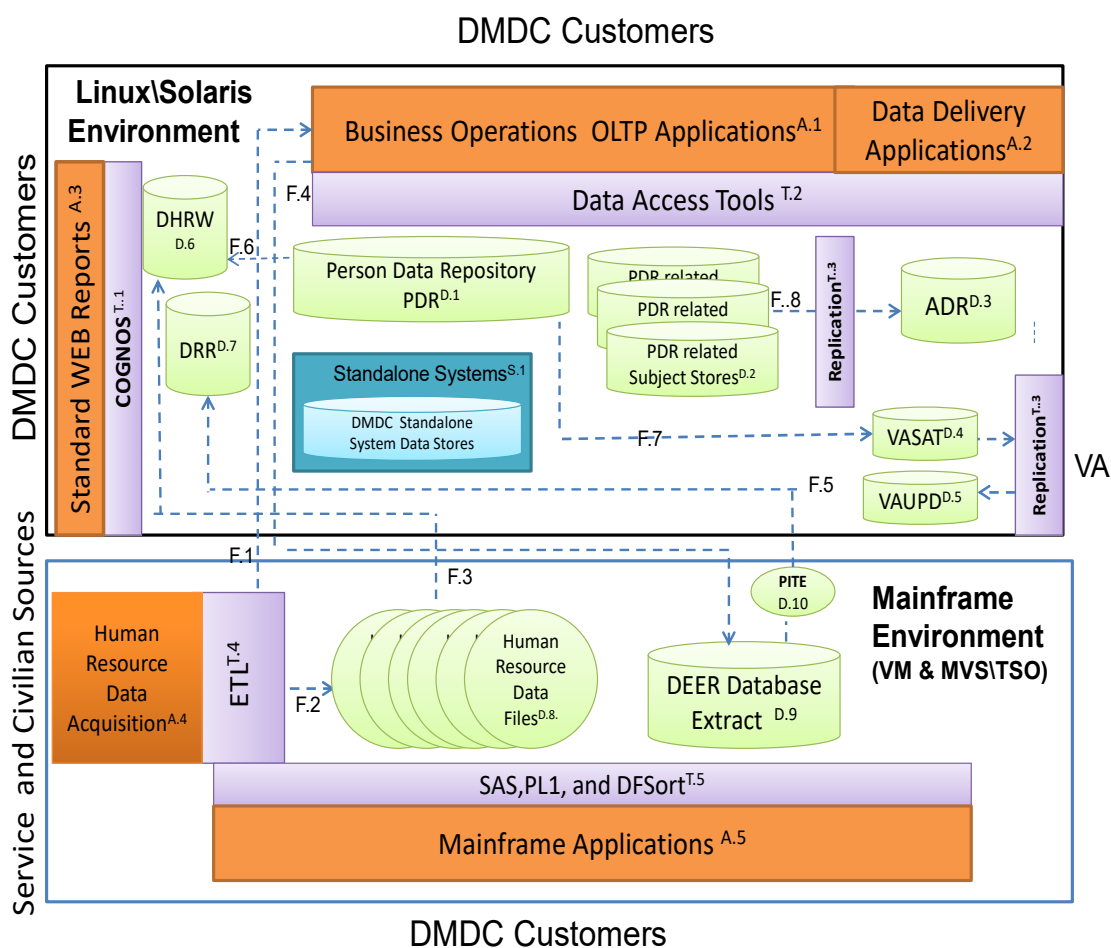
## Current DMDC IT Environment

This document describes the current DMDC IT environment and standards. Figure 1 represents the primary features and platforms within DMDC divided by four aspects of our IT

- Applications
- Data Access
- Data Stores
- Flow of information

Each is followed by a description block for each marked component. An additional section will address large systems developed or migrated to DMDC and how they fit in the overall environment. Many of these are standalone outside the standards provided in this document for new development and are being addressed by a migration strategy. The current standard tools and procedures at DMDC are then described in section 1 through 3

**Figure 1 - Current High-level DMDC Operational Environment**



**Linux\Solaris Environment**

See Section 1 for Hardware Environment

Set Section 2 for Software Environment

**Applications (see Section 3 DMDC Applications Standards)**

- A.1 250 JAVA OLTP Web applications that access data from PDR and PDR-Related Subject DBs via a DMDC data access tools like CUF, DOMAINS EJB (legacy), DOAP (legacy) or by direct access using JDBC or JPA.**
- A.2 200 System to system customers that are registered and secured to receive specific data from DMDC via a standard XML format from a common JAVA Web service called Real-Time Broker Service (RBS) and Batch Broker Service(BBS) accessing ADR. Most customers are outside DMDC but some internal applications and systems use RBS to get person and personnel data.**
- A.3 Java web applications that provide standard reports on personnel, health care enrollment and other HR related data by accessing COGNOS stored reports.**
- A.4 File transfer from Uniformed Service personnel and other DoD sources who send daily, weekly, and monthly transactions to DMDC.**
- A.5 Mainframe applications for data preparation (ETL), data analysis, reporting and distribution of information sent from uniformed service personnel and other DoD sources, in addition to reporting on information provided only by OLTP business operations via the Point in Time Extract (PITE), e.g., personnel family members.**

**Data Access Tools (see section 4 DMDC Data Access tools)**

- T.1 COGNOS is a Reporting tool for creating data stores and cubes and an interface for accessing detailed or summary reports provided by DHRW or DEERS Reporting System (DRS).**
- T.2 Four primary access tools at DMDC - GOTS data access and update frameworks (the new Common update frameworks (CUF) and legacy Domains EJB\ DOAP frameworks) - JDBS, JPA, iBATIS for access and update - RBS \BBS Data Deliver Applications for ADR data store for system to system access only.**
- T.3 Replication is done today with ORACLE Advanced Replication but DMDC is migrating to Symmetric DS.**
- T.4 Today DMDC is doing ETL using SAS and PL1 programs to edit and process files sent to the DMDC mainframe environment by personnel and finance sources. These transactions are stored in files on the mainframe, some are sent to the DHRW DB on LINUX\SOLARIS and all are provided to the PDR as transactions to Business Operations OLTP Applications. We are migrating to use IBM InfoSphere on LINUX\SOLARIS for ETL.**
- T.5 SAS PLI and a sort tool called DFSort are the primary means of data access and processing on the mainframe.**

**Data Stores (see Section 5 DMDC Data Store Standards)**

- D1 Person Data Repository (ORACLE RDBMS) is the largest data store at DMDC (50 million person records) and contains all DoD persons, with their contact and DoD affiliation information. It creates the unique identifiers used for DoD persons (DoD EDI PI) and their Beneficiary roles under a DoD Benefits Number (DBN). PDR serves as the single source of identity for DoD business-related DBs within DMDC and across the DoD.
- D.2 There are data stores (most ORACLE RDBMS, some NOSQL) termed PDR-related Subject stores that are virtualized to PDR common person identity by metadata and JAVA Objects using the Common Update framework I (see Section 2 under the Common Update Framework). Those that are virtualized contain business area data specific to the different DMDC Business Operations (e.g. Health Care, Life Insurance, Education, DoD patient registration, life insurance etc.,). These stores do not store duplicate person, contact or affiliation information but share that data from the PDR.
- D.3 ADR is the Authentication Data Repository (ORACLE RDBMS) a data mart of current information for person from PDR and other PDR-related subject stores. It is accessed by RBS\BBS Java Web services (see Applications Above under A.2).
- D.4 The VASAT (ORACLE RDBMS) is the Veterans Administration Satellite and it is created from PDR military service current and historic information on separation and other factors that affect a servicemen and their family members VA benefits. It is replicated to a data store at the VA called VADIR where it is used for transition processing from DoD to VA.
- D.5 The VAUPD (ORACLE RDBMS) is replicated to DoD from VA's VADIR DB and provides information about benefits and compensation programs administered by VA on our current and prior service members and their families.
- D.6 DHRW (ORACLE RDBMS) is the Personnel Warehouse which stores monthly master and gains and losses transactions from the Personnel Data coming from data acquisition ETL in addition to the MGIB data extracted monthly from the PDR for reporting.
- D.7 DRR is the DEERS Reporting Repository which loads the monthly PITE to do enrollment reporting D.8 Service master files on the mainframe after processing thru ETL.
- D.9 DEERS Database Extract is a VSAM file created by adds and updates to sponsor and family information on a real-time basis from the Database Extract application running against PDR and MEDSAT.
- D.10 Point in Time Extract (PiTE) file created monthly from the DEERS Database Extract on the mainframe.

## Data Transfers

- F.1 Files from Human resource Data Acquisition are sent thru preprocessors on Mainframe to format data for PDR storage. These files include Personnel Gains, Losses, and attribute change feeds sent from the Uniformed Services as well as support information like guard reserve activation and deployment. Monthly master files sent from the uniformed service files are compared month-to-month, to discover additional attribute changes, which are also sent to PDR. Most of these files are processed to PDR via the Personnel and Finance Transfer (PFT) a Business operations OLTP Application using the CUF data access tool.
- F.2 Monthly master files and daily/weekly gains and loss files are stored on the mainframe. Attribute change transactions are only sent in the F.1 data transfer for the PDR.
- F.3 Monthly master and daily/weekly gains and loss files are sent from the mainframe to be stored in the DHRWS (an RDBMS) for reporting.
- F.4 A Data Extraction is run against the PDR and MEDSAT (a Health Care Enrollment subject DB ) as specified data elements change during OLTP processing (using database triggers) for members and their families. These family data are sent to a VSAM file on the mainframe, which is then used to create monthly Point in Time Extract (PITE) files to use in DMDC reporting and sent to selected DMDC customers (e.g., the medical community DHA).
- F.5 Data are extracted as the monthly PITE and sent the DRR to be used in creating reports for display and distribution via COGNOS.
- F.6 Educational benefits and personnel data are extracted monthly from PDR for Montgomery GI Bill and sent to the DHRWS.
- F.7 Personnel data required by VA for transition and benefits are extracted from PDR throughout the day to maintain the Veterans Administration Satellite (VASAT).
- F.8 Data are replicated from the PDR and MEDSAT via triggers that create record changes in the ADR.

## Standalone Systems and Data stores at DMDC

Most DMDC applications and systems utilize the common person, contact and affiliation available from either PDR or ADR and follow DMDC standards for applications. Those that currently do not, fall into one of three categories:

1. DoD Business Systems inherited by DMDC that have not yet been migrated to utilize this common information or any of the common framework or COTS standards DMDC has put in place.
2. Those that, while following DMDC standards, are not person entity -based (e.g. other entities like organizations).
3. Those applications developed at DMDC before the standards were established or, while following standards:
  - Neither share in the common person data, nor
  - Make their subject area information available across DMDC applications.

The issue of differing platforms and IT standards is addressed by either migrating the system to common standards or adoption of the platform into the common standards. The issue of providing common person data and sharing of Subject data (when applicable) is addressed for categories 1 and 3 above by a migration strategy:

The first step in migration is to store the Common Person ID (DoD EDI PI) in the person based standalone data store so that the data store may be interoperable with other DMDC data stores.

The next step is connecting the application / system to PDR CUF framework to access and, when applicable, provide updates to common person data.

The final step is becoming a PDR-related subject data store where the subject data can be accessed with the person data under a common object model via a single inquiry. The deciding factors for this are:

- A requirement to share that subject data with other parts of DMDC,
- An advantage for configuration management afforded by the common CUF lexicon maintaining the relationship between applications and the data stores and entities they use across DMDC.

## Section 1 – DMDC Hardware Environment

### Linux\Solaris

1. Oracle (Exadata, SUN Servers)
2. Cisco Network Devices
3. Hitachi Storage
4. NetApp Storage

## Section 2 – DMDC Software Environment

### Linux\Solaris

1. Sunview Software - ChangeGear
2. VMWare (vRealize Suite, vSphere with Operations Management, vCloud Suite, Horizon 6, Orchestrator)
3. Oracle (Database, Webserver, Identity Manager, Records Management, LDAP)
4. RedHat Linux
5. Microsoft (Office Suite, SQL, OS, SharePoint)
6. SolarWinds (All Modules)
7. Puppet Labs

## Section 3 – DMDC Application Standards

### Linux\Solaris

1. Applications are written in JAVA employed on ORACLE WEBLOGIC (moving to JBOSS)
  - 1.1. Internal Shared Component/Service
  - 1.2. Bundled Shared Component (Java Archive JAR)
  - 1.3. Shared Library Component (Enterprise Archive EAR, Web Archive WAR,
  - 1.4. Internal Web Service (JAXWS, Restful, XML etc..).
2. JAVA Application Types
  - 2.1. Browser-Based Web UI
  - 2.2. Web Servers (system to system)
  - 2.3. Batch processing
3. Browser-Based Web UI
  - 3.1. DMDC Standards
    - 3.1.1. JSF
    - 3.1.2. SpringMVC/WebFlow
    - 3.1.3. Javascript not sure what that is going to be
    - 3.1.4. Struts (current but deprecated)
    - 3.1.5. Portlet UI (current but deprecated)
  - 3.2. Application Security (figure 2 area A.1)
    - 3.2.1. Operator- based applications - requiring provisioned users for access

- 3.2.1.1. Operator Authentication requires PKI issued on a certificate residing on the following tokens:
  - 3.2.1.1.1.1. DoD issued Common Access Card (CAC)
  - 3.2.1.1.1.2. Registered PIV card from other federal agencies
  - 3.2.1.1.1.3. Registered PIV-I card from industry and other sources
- 3.2.1.2. Authorization for Operator-based applications
  - 3.2.1.2.1.1. EMMA(GOTS) is used to provision users to DMDC applications
  - 3.2.1.2.1.2. AS-IS (GOTS) for Operator Consent/Authentication is used by DMDC applications to access provisioned data on EMMA
- 3.2.2. Self-service applications - DoD beneficiaries accessing their own information
  - 3.2.2.1. Authentication
    - 3.2.2.1.1.1. DoD issued Common Access Card (CAC)
    - 3.2.2.1.1.2. DSLOGON Web service (GOTS) for authenticating a self-service account id and password for non-CAC holders
  - 3.2.2.1.2. Authorization
    - 3.2.2.1.2.1. The DoD ID Number (DOD EDI PI) returned from the Authentication process is used to restrict access to the authenticated persons' information.
- 3.3. Web Server (System to system) applications standards (figure 2 area A.1).
  - 3.3.1. Application Standards:
    - 3.3.1.1. WSDL-based Web Service (JAXWS).
    - 3.3.1.2. XML/JSON Restful (JAX-RS / Jersey).
    - 3.3.1.3. Plain non-WSDL-based XML.
  - 3.3.2. Security Standards:
    - 3.3.2.1. Transport Layer PKI System to System Security (DoD issued system certificates).
    - 3.3.2.2. VPN System to System Security.
- 3.4. Batch Applications
  - 3.4.1. Continuous Batch – a standard Events framework is available to drive the process of change events within the data stores (using triggers to detect and write change events to work tables). These events can take the form of:
    - 3.4.1.1. Sending a notification or correspondence to a system or a person either by transactions, email or postal mail.
    - 3.4.1.2. Driving redetermination of derived information create by JAVA applications using a COTS rule engine (PEGA) which apply DoD rules and regulations to changes in person and affiliation information. This derived information can be a person's DoD benefits and entitlements (e.g. Health Care Coverage, ID card re-issuance or termination, etc.) or other information like active duty status eligibility based information.
  - 3.4.2. File Batch - A standard Batch Application Framework is provided that accepts files as input and tracks the processing, providing automatic restart if the applications fails mid-process.
  - 3.4.3. Quartz (open Source) based Batch.

- 3.4.4. 3.4.1 and 3.4.2 are performed within a standard batch or event scheduler framework (see CUF).
- 4. Logging and Auditing standards:
  - 4.1. JMS Queue based Transaction Recorder API.
  - 4.2. ELK based LogMonitor (Elasticsearch, Logstash, Kibana).

## Mainframe

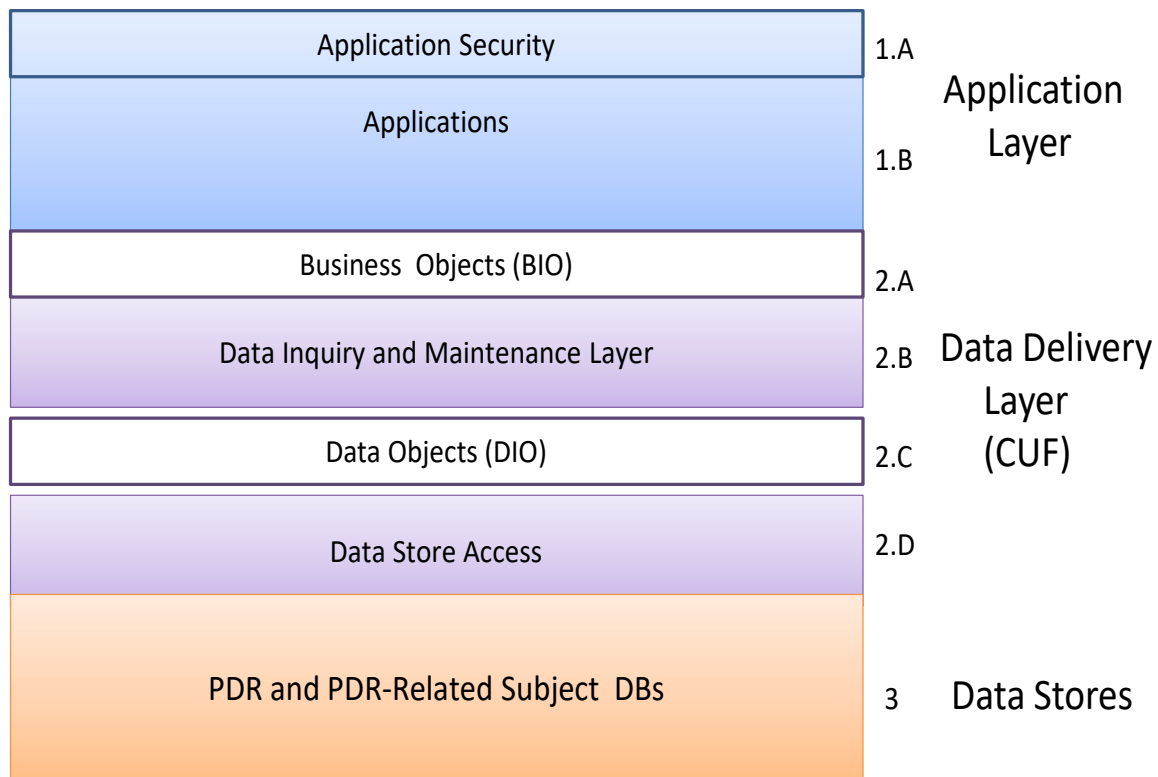
- 1. PL1 applications are generally used to prepare data (ETL)
- 2. SAS and DFSort are used for reporting and analysis

## Section 4 - DMDC Data Access Standards

### Linux\Solaris

- 1. There are five database access methods used within DMDC data access frameworks :
  - 1.1. Spring JDBC
  - 1.2. JPA (EclipseLink)
  - 1.3. MyBatis
  - 1.4. JDBC (current but deprecated)
  - 1.5. DMDC Core (custom JAVA with PLSQL packages) (current but deprecated)
- 2. DMDC Application Data access Frameworks
  - 2.1.1. DMDC RBS\BBS Web service when used by DMDC Applications layer to ADR
  - 2.1.2. DMDC Common Update Framework (CUF) to PDR and PDR-related subject stores
  - 2.1.3. DMDC Domains EJB (current but deprecated)
  - 2.1.4. DMDC DOAP framework (current but deprecated)
- 3. RBS\BBS - this is a JAVA xml metadata driven access layer to obtain attributes from ADR data mart (which stores person and current personnel and other related data replicated from PDR)
  - 3.1. RBS - Real-time Broker Service
    - 3.1.1. Primary mission has been in support of logical access control systems. It provides rapid response to DoD Web applications to supply ABAC and registry inform for DoD user provisioning.
    - 3.1.2. It has also served as a Policy Decision Point which applies customer defined rules to return access decisions (Y/N) and other business-rule oriented inquiries.
    - 3.1.3. It is used internally by Business Operational inquiry applications in some cases:
      - 3.1.3.1. Appropriately to extend its mission of logic access support to Physical Access Control-based internal Systems (iMESA and DBIDS).
      - 3.1.3.2. In some cases, because the CUF framework was not yet available to PDR (these applications are being moved to CUF).
  - 3.2. BBS - Batch Broker Service
    - 3.2.1. Its primary mission has been data distribution in support of other DoD systems (e.g. IdSS under DISA for CAC holders identity and credential information, Marine Corps personnel system to send family member information, etc.)

- 3.2.2. BBS is a *publish-and-subscribe* Web site.
- 3.2.3. Rules are defined for populations of interest and attributes required.
  - 3.2.3.1. Internal flags are stored for the customers based on adds or changes within their defined population of interest for selected attributes as they are modified in ADR when new information is replicated from PDR
  - 3.2.3.2. Customers request these updates to their persons of interest periodically to the BBS Web site
- 3.2.4. As with RBS this BBS service is sometimes used internally in DMDC to distribute data which can be eliminated as we enhance our capabilities to provide access to PDR and business area DB data to be more flexible to non-person based inquiries
- 4. Common Update Framework (CUF) a GOTS which simplifies access for greater agility in applications development and maintains data integrity by applying Common and consistent data quality rules across all applications. CUF virtualizes our PDR and PDR-related subject data stores into a single lexicon of accessible Business objects for our diverse applications.

**Figure 2 – Application Framework**



#### 4.1. Common Update Framework ( Figure 2 – 2.A. thru 2.D )

- 4.1.1. CUF itself is a virtual framework that consists of: the CUF controller, application interface Business Objects (BIOs), data store interface objects (DIOs) accessed via the Data Access Processors, Subject Processors which control inquiry and maintenance of all BIOs to and from related DIOs.

This separation of Data Objects, representing the data store models and the BIOs representing the view of the data from a business application and client standpoint provides the flexibility to represent data as the business requires it without regard to how or where it is store. This level of separation of source from usage allows us to change how and where we store data without affecting the applications or user community.

- 4.1.2. **The CUF controller** using the BIOs sent from the application

- 4.1.2.1. Instantiates the related subject processors, and data access processors
  - 4.1.2.2. Orchestrates these JAVA components using the hierarchy of BIOS and information stored in the metadata of CUF

- 4.1.3. **Business Interface Objects (BIO)** on all person and PDR-related subject data are available in the CUF across PDR and all PDR-related subject stores in a single hierarchy of JAVA Objects beginning with PERSON and spanning all related data stores This relates and virtualizes the data from many data stores into a single object model for access and update at DMDC

- 4.1.3.1. The hierarchy, definitions, and attributes for all BIOS are stored in metadata called the LEXICON where the relationship of the BIO to a specific data store and entity are also kept
  - 4.1.3.2. Requests for access and maintenance of each BIO is stored in application privilege metadata called the permissions store
  - 4.1.3.3. BIOs are delivered to and from the JAVA application layer via an EJB that contains Business Interface Objects (BIOs) transmitted via XML (2.A)

- 4.1.4. **A CUF client application interface** is created as a Jar file for an applications once it is recorded in the CUF Metadata. Here also information is provided by the application owner pertaining to what BIOS the application requires and the applications operating characteristics (e.g. business context, GUI\system to system\batch application, etc.,).

- 4.1.4.1. **Inquiries** are performed via the CUF client by sending the CUF specific BIOs required with person identity information for the person or family required. Data will be returned in BIOs by the CUF for use by the application
  - 4.1.4.2. **Updates and Adds** are done by the applications by sending BIOs via the CUF client to the CUF server in push mode.

- 4.1.5. **Subject Processors** - are written to convert information from data store format to business client format and most importantly for data quality to ensure that data are valid,

complete and maintained correctly. Subject processors are grouped into libraries based on DMDC customer and their business-area information maintenance requirements.

- 4.1.6. **Data Access Processors** - each data store has a JAVA data access processor written to access it. This Data access processor which is recorded in the lexicon along with the data objects it provides:

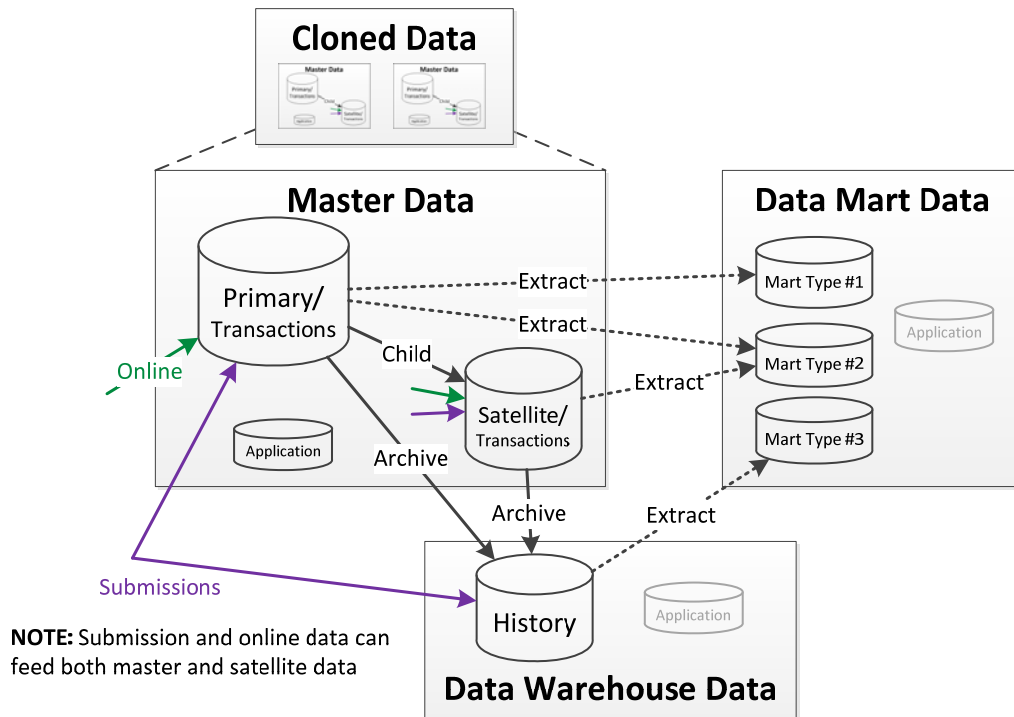
- 4.1.6.1. The requirements for a data access processor under CUF is to provide JAVA Data Objects that can be recorded and related to subject processors in the Metadata for CUF (Lexicon).
- 4.1.6.2. For RDBMS data store access can be performed using the data access technologies under list in Section 4 above

## Section 5 - DMDC Data Store Standards

### Database Platforms

#### Linux\Solaris

1. Most of our data stores are RDBMS using ORACLE.
2. NOSQL databases are present products like MONGO and Cloudera

**Data Store Definitions:****Data Store Type Definitions****Master Data:**

Master data is atomic, authoritative data.

- Primary:** Primary data is atomic business data. Depending on business need, subsets of master data can be archived, or snapshots of master data can be taken to create history data.
- Satellite:** Satellite data is atomic business data. Satellites are built to logically separate groups of functionally independent master data. This separation can be used to offload transactions and/or create independent operation from related primary data. Like it's parent master data, satellite data can be archived or snapshots taken to create history data.
- Transaction:** Transaction data is atomic data that may be business data (claims transactions) or relate to business data (audit or performance capturing transactions).
- Application:** Application data is used by application(s) to function. Since applications run using primary, mart, or warehouse data, application data exist with primary, mart, or warehouse data.

**Cloned Data:**

Cloned data are read only, full copies of Master data that offloads work from master data sources.

**Data Mart Data:**

Data mart data is derived data that offloads work from other sources, allows independent operation from other sources, or creates denormalized data for easier use. It contains subset data for a specific business purpose. It can contain aggregate and/or detail data and be sourced from master or warehouse data stores.

**Data Warehouse Data:**

Data Warehouse data contains full history, including all transactions and snapshots of master data that must be maintained over time. Warehouse data is gathered from all aspects of the business so that reporting can be done across areas or lines of business. In the case of submission data, DMDC warehouses snapshots and transactions of service personnel system master data.

**NOTE:** These data store type definitions are provided to clarify types of storage implementation that can be used to satisfy requirements. DMDC does not have to implement all of them.

# **DMDC IT Environment and Standards**

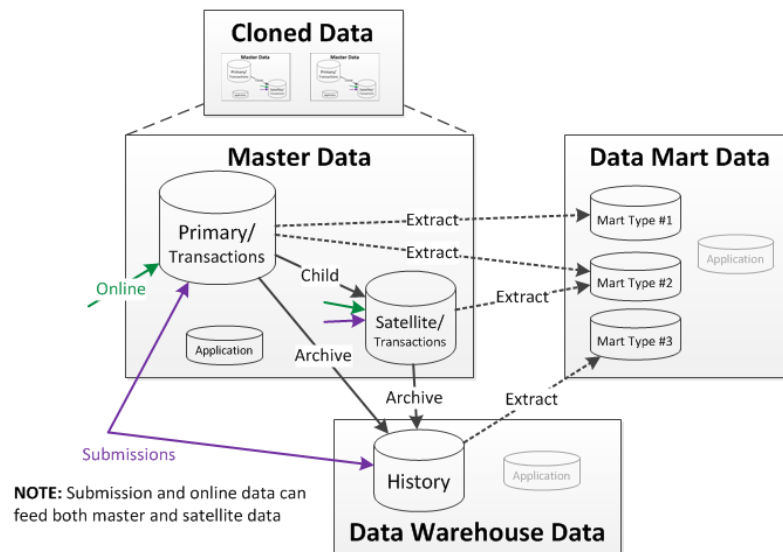
## Data Architecture Requirements Model:

	DATA COLLECTION	DATA STORAGE	DATA USE
<b>REQUIREMENTS</b>	<ol style="list-style-type: none"> <li>1. Process Batch Data (<b>Use Case</b>)</li> <li>2. Process Real Time Data (<b>Use Case</b>)</li> <li>3. Process Online Data (<b>Use Case</b>)</li> <li>4. Process Ad-Hoc Data (<b>Use Case</b>)</li> <li>5. Use a common framework for each use case (<b>Rqt</b>)</li> <li>6. Use common services for critical functions (<b>Rqt</b>) <ul style="list-style-type: none"> <li>- Identity - Authentication/Access</li> <li>- Audit - Meta data (Valid Values)</li> <li>- Reference data (CTRY, UIC, etc.)</li> </ul> </li> </ol>	<ol style="list-style-type: none"> <li>1. Use master data as preferred choice for all uses cases (<b>Rqt</b>)</li> <li>2. Partition Master Data (create satellites) (<b>Rqt</b>) <ul style="list-style-type: none"> <li>- For ease of use by business</li> <li>- To implement security control</li> <li>- For performance/scale</li> </ul> </li> <li>3. If full data history is needed, store it in a warehouse (<b>Rqt</b>)</li> <li>4. If large master data sets require complex joining, store it in a warehouse (<b>Rqt</b>)</li> <li>5. Build data marts when data must be derived (aggregated or denormalized) for reporting efficiency (<b>Rqt</b>)</li> <li>6. Minimize extract and archive complexity (<b>Rqt</b>)</li> <li>7. Use the same storage technology for data stores (<b>Rqt</b>)</li> <li>8. Use the same technology to extract/archive (<b>Rqt</b>)</li> </ol>	<ol style="list-style-type: none"> <li>1. Perform manual inquiry (<b>Use Case</b>)</li> <li>2. Exchange Data <ul style="list-style-type: none"> <li>- Low Volume (<b>Use Case</b>)</li> <li>- High Volume (<b>Use Case</b>)</li> </ul> </li> <li>3. Reporting <ul style="list-style-type: none"> <li>- Business Analytics (<b>Use Case</b>)</li> <li>- Big Data Queries (<b>Use Case</b>)</li> <li>- Simple Queries (<b>Use Case</b>)</li> <li>- Decision Support (<b>Use Case</b>)</li> <li>- Detail (roster) reports (<b>Use Case</b>)</li> </ul> </li> <li>4. Distribute Data <ul style="list-style-type: none"> <li>- Provide data to disconnected systems (<b>Use Case</b>)</li> <li>- Provide manual extracts (<b>Use Case</b>)</li> <li>- Provide recurring extracts (<b>Use Case</b>)</li> <li>- Personal notification (<b>Use Case</b>)</li> </ul> </li> <li>5. Use a common framework for each use case (<b>Rqt</b>)</li> </ol>

**LEGEND**

**Use Case** – Use cases are functional capabilities DMDC provides.

**Requirement (Rqt)** – Requirements are conditions DMDC requires when delivering a use case capability.



**Data Architecture Use Cases:**

A use case defines the interactions between external actors and the system under consideration to accomplish a goal. The use cases in this document capture a comprehensive, conceptual list of the interactions operator, user, and system to system interactions that DMDC customers have with DMDC data systems.

**Data Collection**

Data Collection use cases are the functional interactions that result in DMDC receiving data.

Use Case	Actor Type	Description
Submit Batch Data	System to System	External systems compile records and send them on a recurring basis to a DMDC System to process asynchronously.
Submit Real Time Data	System to System	External systems submit one or more records to a DMDC System in real time to process synchronously.
Process Online Data	Operator/User	Operators and users transact within a DMDC system to write data. The DMDC system may also support some read transactions as part of the business processes used to collect data.
Process Ad-Hoc Data	Operator	A customer provides an ad-hoc data extract to a DMDC operator who stores the data for use.

**Data Storage**

There are no Data Storage Use Cases. There are storage requirements

**Data Usage**

Data usage use cases are the functional interactions where DMDC customers receive data.

Use Case	Actor Type	Description
Perform manual inquiry	Operator/User	An operator or user performs inquiries in a DMDC system that only supports read only transactions.
Exchange low volume data	System to System	An external system retrieves real time or batch data in volumes that can be supported by a primary data store.
Exchange high volume data	System to System	An external system retrieves real time or batch data in volumes that cannot be supported by a primary data store.
Run business analytics queries	Operator	An operator uses Business analytics software to develop business performance insights of based data and statistical methods.
Run big data queries	Operator	An operator runs queries that require intensive computing resources, very large sets of data, and/or complex joins of data that exceed the capacity of normal primary data stores.
Run simple queries	Operator	An operator runs queries whose resource requirements, amount of data, and/or complexity can generally be supported by the capacity of normal primary data stores.
Run decision support reports	Operator	An operator runs reports of aggregate data.
Run detail reports	Operator	An operator runs atomic (non-aggregated data) detail reports.
Provide data to disconnected systems	System to System	A system provides data to separate systems to allow independent operation.
Provide manual data extracts	Operator	An operator provides an ad-hoc snapshot set of data to a customer.

Use Case	Actor Type	Description
Provide recurring data extracts	Operator	An operator provides a recurring snapshot set of data to a customer.
Personal Notification	System to System	Send a data driven message to a person.

**Data Architecture Requirements:**

Requirements describe a condition to which a system must conform; either derived directly from user needs. The architectural requirements defined in this document define those architecturally significant, data related conditions that DMDC requires to support the use cases in this document.

**Data Collection**

Apply the following requirements when building systems to support data collection use cases.

Requirement	Rationale
Use a common framework for each data collection use case	Each data collection use case has discreet aspects to design. In principle, DMDC should have one standard framework (design pattern, technology, etc.) for each use case. Every additional design that gets implemented has overhead such as system support, development skillset, and licensing.
Use common services for critical functions required to collect data: - Identity- Audit - Authentication/Access - Meta data (Valid Values) - Reference data (CTRY, UIC, etc.)	There are certain functions that are common to all data collection use cases. In principle, DMDC should have one standard service (design pattern, technology, etc.) that can be used by every system to perform those functions. Every additional design that gets implemented has overhead such as system support, development skillset, and licensing. Additionally, if multiple services are implemented, there is significant risk that they may behave inconsistently, resulting in data quality issues and other anomalous behavior.

**Data Storage**

Apply the following requirements when building data stores into which data is collected and from which it is used.

Requirement	Rationale
Use master data as preferred choice for all uses cases	Warehouses, data marts and data clones have implementation and ongoing support costs above and beyond the costs to implement and maintain the primary systems and their data stores. Use Master data stores if at all possible.
Avoid derived data in master data stores	If reports can be effectively serviced from a master data store, do so. If reporting efficiency requires derived data (aggregation, sub setting, or denormalization), segregate that data into a data mart.
Partition Mater Data (create satellites)	Logical and physical data partitioning should be done to support business needs. Logical partitioning puts related data into separate stores (schemas) to separate business use and can also support security approaches. Logically separated data can be collocated with its parent primary data. Physical partitioning primarily separates data for performance, and may also be done for business use or security reasons. Physically partitioned data needs to be given separate resources (i.e hardware) to realize performance benefits.
If full data history is needed, store it in a warehouse	Full history data (like monthly snapshots or change transaction from heavy use master data stores) tend to get large. A Warehouse solution should be implemented with technology optimized to work with that data.
If large master data sets require complex joining, store it in a warehouse	Some types of queries (reporting or analytics) are sufficiently complex that they won't operate effectively against even moderately sized primary data source. A Warehouse solution should be implemented with technology optimized to work with that data.

Requirement	Rationale
Minimize extract and archive complexity	To the greatest extent possible, define a simple, repeatable extract and archive processes and keep a central track of them. Historically, project teams have had issues identifying dependencies when working with data because there are multiple ways which data is extracted and moved between data stores. These have resulted in project delays because those processes are identified at the last minute and SRTs because the dependencies were never identified until something went wrong in production.
Use the same storage technology for data stores	DMDC uses many vendor products that provide the same basic capability, such as Oracle, Exadata, MySQL, and SQL Server for RDBMS. We have looked at multiple NoSQL technology, such as Couch, MongoDB and Hadoop. Each has additive system support and licensing costs. Each new technology stretches our systems staff, reducing their ability to service our mission requirements. While different use cases may rally be better suited to different technical solutions, we should strive to limit the number of technologies we use.
Use the same technology to extract/archive	DMDC uses multiple solutions to extract and archive data (many home grown). The same costs associated with supporting data storage technology also apply to ETL/Archive technology

### Data Usage

Apply the following requirements when building systems to support data usage use cases.

Requirement	Rationale
Use a common framework for each data collection use case	Each data usage use case has discreet aspects to design. In principle, DMDC should have one standard framework (design pattern, technology, software toolset for manual uses, etc.) for each use case. Every additional design or software toolset that gets implemented has overhead such as system support, personnel skillset training, and licensing.